A Service-Oriented Platform for Interactive 3D Web Mapping

Jan Klimke, Benjamin Hagedorn, Jürgen Döllner

Hasso-Plattner-Institut, University of Potsdam

{jan.klimke, benjamin.hagedorn, doellner}@hpi.uni-potsdam.de

Biography

Keywords: Service-oriented 3D Mapping, 3D Map Creation, 3D Map Delivery, 3D Map Styling, 3D City Models

Abstract: Design, implementation, and operation of interactive 3D map services are faced with a large number of challenges including (a) processing and integration of massive amounts of heterogeneous and distributed 2D and 3D geodata such as terrain models, buildings models, and thematic georeferenced data, (b) assembling, styling, and rendering 3D map contents according to application requirements and design principles, and (c) interactive provisioning of created 3D maps on mobile devices and thin clients as well as their integration as third-party components into domain-specific web and information systems. This paper discusses concept and implementation of a service-oriented platform that addresses these major requirements of 3D web mapping systems. It is based on a separation of concerns for data management, 3D rendering, application logic, and user interaction. The main idea is to divide 3D rendering process into two stages. In the first stage, at the server side, we construct an image-based, omni-directional approximation of the 3D scene by means of multi-layered virtual 3D panoramas; in the second stage, at the client side, we interactively reconstruct the 3D scene based on the panorama. We demonstrate the prototype implementation for real-time 3D rendering service and related iOS 3D client applications. In our case study, we show how to interactively visualize a complex, large-scale 3D city model based on our service-oriented platform.

Introduction

The availability of 3D geodata, its volume and its quality are constantly growing. Therefore, a high quality 3D visualization of massive and detailed data sets represents a computationally expensive task: It consumes large amounts of main memory, disk, network, CPU, and GPU resources in order to deliver an interactive user experience and a good visual quality. For distribution of such 3D contents in a web environment, two principal concepts can be found in existing 3D web mapping approaches, *client-side 3D rendering*, i.e., assembly and streaming of 3D display elements such as 3D geometry and textures by a 3D server, whereby the corresponding 3D clients manage scene graphs and perform real-time 3D scene rendering (e.g., Google Earth, OGC W3DS); and *server-side 3D rendering*, i.e., assembly and rendering of 3D display elements by the 3D server, which delivers views of 3D scenes to lightweight clients (e.g., OGC WVS).

Taking into account the increasing complexity and size of geodata for 3D maps (e.g., complex 3D city models), the need for high-quality visualization (e.g., illustrative or photorealistic rendering), and rapidly growing use of mobile applications (e.g., smart phones and tablets), these principal concepts are faced by fundamental limitations: client-side 3D rendering fails to provide fast access to massive models due to bandwidth limitations and cannot guarantee high-quality rendering results as the graphics capabilities of nearly all mobile devices differ significantly, while a pure server-side 3D rendering is inherently limited with respect to interactivity and does not take advantage of today's mobile device graphics capabilities.

In this paper, we present a service-oriented platform for interactive 3D web mapping based on a separation of concerns for data management, 3D rendering, application logic, and user interaction. The main idea, in a nutshell, is to divide the 3D rendering process into two stages. In the first stage, at the server side, we construct an image-based,

omni-directional approximation of the 3D scene by means of multi-layered virtual 3D panoramas; in the second stage, at the client side, we reconstruct the 3D scene based on the panorama. This approach profits from both: The high quality, server-side image generation and the increasing graphics capabilities of client devices.

Background and Related Work

The interoperability of systems and applications dealing with geodata is an central issue in order to build systems out of interoperable software components for geodata access, processing, and visualization.

Beside a common understanding on information models (Bishr, 1998), definitions of service interfaces are necessary. The Open Geospatial Consortium (OGC) defines a set of standardized services, models, and formats for geodata encoding and processing. For example, a Web Feature Service (WFS) (Vretanos, 2005) can provide geodata, encoded in the Geography Markup Language (GML) (Portele, 2007) or City Geography Markup Language (CityGML) (Gröger et al., 2012), and processed by a Web Processing Service (WPS) (Schutt, 2007).

For geovisualization processes a general portrayal model is provided by the OGC that describes three principle approaches for distributing the tasks of the general visualization pipeline between portrayal services and consuming applications (Altmaier and Kolbe, 2003; Haber and McNabb, 1990). While the OGC Web Map Service (WMS), providing map-like representations of 2D geodata, is widely adapted and used, 3D geovisualization services have not been elaborated to a similar degree. Several approaches for 3D portrayal have been presented (Basanow at al., 2008). Two types of 3D portrayal services, currently discussed in the OGC context, can be distinguished:

- *Web 3D Service* (W3DS) (Schilling and Kolbe, 2010): It handles geodata access and mapping to renderable computer graphics primitives (e.g., textured 3D geometry represented by scene graphs) and their delivery to client applications.
- *Web View Service* (WVS) (Hagedorn, 2010): It encapsulates the image generation process (i.e., it implements the geovisualization pipeline) of 3D models, delivering rendered image representations ("portrayals") to client applications.

By focusing on developing international standards for 3D Portrayal, an interoperable, service-based system can be built, that allows replacing component implementations selectively with other implementations. Further, system components, especially a 3D rendering service or a W3DS instance can be reused in other systems.

Several approaches exist for remote rendering of 3D virtual environments (Boukerche and Pazzi, 2006; Paravati et al., 2011). Mostly, they rely on constant transmission of single images (Wessels et al., 2011) or video streams (Lamberti and Sanna, 2007; Pajak et al., 2012) to client applications. In contrast to those applications that are completely dependent on the current data throughput of the network connection, our approach uses a latency hiding technique. This technique allows for a continuous user interaction operating on the locally available data also in situations with very low data transmission rates between 3D rendering server and clients.

Another approach is to manage and render a low resolution 3D model on client side, to allow users to explore the model interactively; when interaction stops, remote rendering is used to create and display an image of the high-resolution 3D model (Koller et al., 2004). For delivery of, e.g., large-scale textured 3D city models, this approach is not suitable, since a transmission and rendering of low resolution model representations on client-side would still exceed network and client-side rendering capabilities. In approach an image-based 3D approximation of a model is created by the client, using image data transmitted from a 3D rendering service.

3D Web Mapping Requirements

Design, implementation, and operation of interactive 3D map services are faced with a large number of challenges including:

(a) Data processing and integration: Massive amounts of heterogeneous and distributed 2D and 3D geodata such as terrain models, buildings models, and thematic georeferenced data form the basis for 3D maps. This data has to be processed for visualization and integrated into the overall visualization system in order to be combined in a 3D map.
(b) 3D map content assembly, rendering and styling: To communicate spatial and georeferenced information, 3D map content must be selected, styled and rendered according to application requirements and design principles.
(c) Interactive provisioning: Created 3D maps should be available on mobile devices and thin clients in an interactive manner. Further, third-party vendors should be able to integrate 3D maps as components into their domain-specific web and information systems.
(d) Interoperability: A system for 3D web mapping should rely on established standards in terms of data formats and service interfaces.

In particular, challenge (c) influences the way a service-oriented system for 3D mapping can be built. Large amounts of data have to be transmitted, processed, and stored for generating a useful 3D map, common client applications have to scale with the size of the underlying 3D model, because they deal with the massive amounts and the complexity of model data on client side. In order to provision 3D maps on a large variety of devices with heterogeneous hardware and software capabilities and undefined as network connection speed, the effort of processing, transfer, and rendering of 3D map content should be decoupled from a client application, while still providing a user with an interactive user experience. This diversity and performance considerations is especially an issue, when designing applications for mobile devices or browser-based 3D mapping solutions. Clients for these platforms should provide a equally high visual quality, regardless of the capabilities of the individual device or platform.

Access to a variety of 3D content and its task and user specific styling is crucial when using a 3D map application. A 3D web mapping system should therefore provide an interface to configure the 3D map in a way that allows to adjust data selection and styling per data set. Further, from a service-oriented point of view, service interfaces should rely on standards, as far as they exist, to enable implementation reuse and interoperability between different implementations of the same service.

A Service-Oriented Platform for 3D Web Mapping

In order to address the requirements mentioned above, we propose a service-oriented decomposition of the 3D mapping process into independent stages: Server-side data processing and 3D rendering and client-side interactive provisioning of 3D contents (Figure 1). In contrast to common approaches for remote 3D visualization of 3D geodata, our approach is based on transmission of images in standard image formats. The complexity of geodata management and 3D rendering is encapsulated on server side. A 3D client uses a Web View Service interface, provided by the 3D rendering server, to request an image-based, omni-directional approximation of the 3D scene by means of multi-layered virtual 3D panoramas (Figure 2). Based on these panoramas, a 3D client reconstructs a lightweight 3D representation of the server-side data.

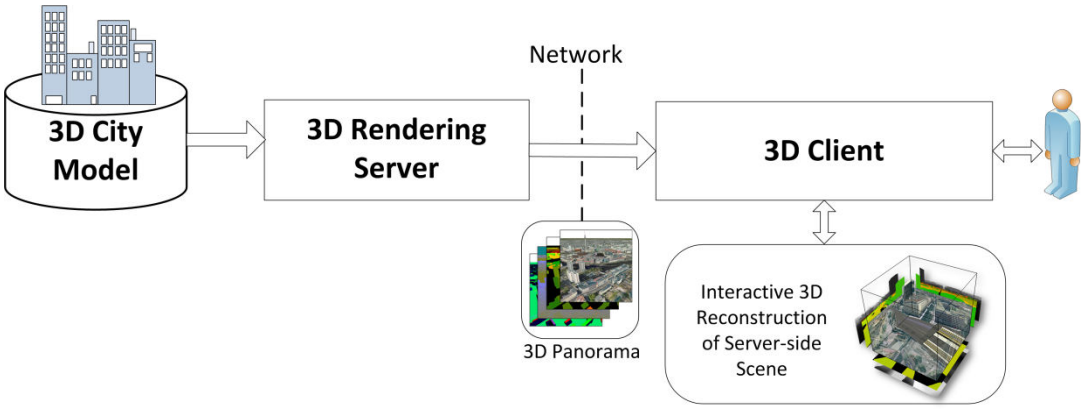

Figure 1: Working principle of the system for interactive 3D mapping. Handling of complex, massive 3D geodata is encapsulated on server side, while clients reconstruct the 3D environment using images generated by a 3D rendering server.

Our system architecture offers a number of key properties that facilitate construction and deployment of interactive 3D web mapping solutions.

- *Scalability*: The approach allows for visualizing virtually arbitrary complex, server-side 3D models on clients by decoupling the complexity of the 3D model data from the complexity of streamed data and therefore rendering complexity on client side: regardless of the model complexity, the streamed virtual 3D panoramas are of fixed size.
- *Configurability*: Features and styles of 3D web mapping can be requested and configured by style parameters in the interface. For example, new stylization techniques can be provided by the 3D server for existing clients without any redeployment or client-side installations.
- *Stability*: The implementation of the core 3D rendering process is based on the a-priori known server hardware and software. In particular, advanced real-time 3D rendering techniques, such as stylization and illustrative techniques, can be implemented in a stable and efficient way. Since image-generation of 3D geodata is performed completely on server side,
- *Robustness*: A client always operates on its cached virtual 3D panoramas. Therefore, the client can guarantee full interactivity regardless of data transmission latency.
- *Extensibility*: Application-specific extensions can be integrated by additional information layers in the virtual 3D panoramas and by additional client-side functions. Examples include selectable model elements (e.g., planning variants in 3D city models), rendering styles per layer, image-based stylings, or placemarks (predefined scene views).
- *Interoperability*: The rendering service implements the Web View Service (WVS) interface for image-based 3D portrayal of Geodata, which currently in the standardization process within the OGC. Further, we support CityGML as standard format for 3D city models through our preprocessing component.

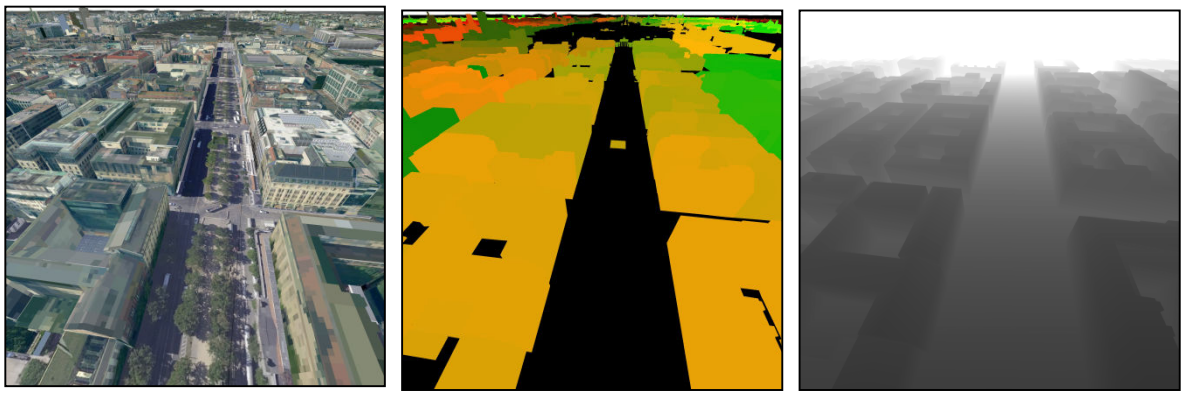

Figure 2: Different image layers of the same view. From left to right: Color image, object id image, depth image.

Architecture

The core component of our architecture is an image-based 3D rendering service that adheres to the Web View Service (WVS) specification and represents a 3D portrayal service that is currently in the standardization process in the OGC. It provides an extendable interface to retrieve rendered images of a 3D scene. It is used by corresponding client applications to request different image layers (e.g., depth or object id of visible image pixels) encoded in standard image formats. The rendering service encapsulates a server-side visualization pipeline. All processing, filtering, styling, and rendering of 3D map contents are hidden from clients and can be controlled through the extensible WVS interface.

Since, massive 3D geodata is not suitable for efficient 3D rendering in its original form, a preprocessor component exists for converting 3D geodata formats, such as CityGML, into a structure optimized for rendering. This structure then holds a compactified representation of geometry and textures that is specific for the techniques for texturing, styling, and object identity management of the rendering service. The preprocessing service represents an independent piece of functionality and could also be implemented as web processing service. The preprocessor is able to convert a range of different formats for 3D models to the same optimized representation. This enables the rendering service to support rendering of models encoded in a wide range of different 3D formats.

Besides the optimized representation, CityGML source data is kept accessible in a database together with the association information between rendering ID (object ID) and CityGML feature identifier. In this way, the system is able to deliver feature data for each rendered object. Thematic data for 3D content that was integrated into the visualization process can be accessed externally by using corresponding Web Feature Services.

A client application implements the user interface. It provides features for both: Camera interaction and query of feature data. A client application works with the image data provided by the rendering service. Due to the different information encoded in image layers (color, depth, object ID, and normals) clients may provide far more possibilities to users compared to a server-side rendering of color information only.

Application specific data is managed through an application config service. This service delivers configuration documents for the client applications, defining, e.g., available model layers, layer-style combinations, image styles, as well as relevant camera positions and orientations for the specific use case. These documents are defined using a web-based administration interface. This additional indirection for the configuration of client applications allows using the same application for different use cases without touching the actual client code.

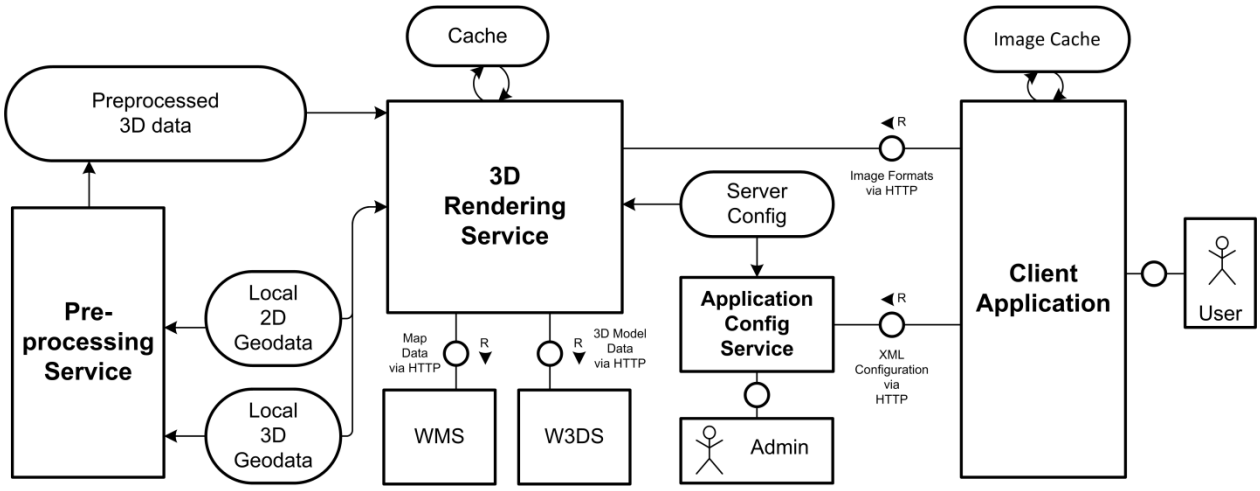

Figure 3: Architecture diagram of the service-oriented platform for web mapping.

Data Integration

Several possibilities for integration of different data types exist on server side as well as on client side. Besides a 3D base model, i.e. a textured 3D city model, other 3D model data, e.g., for visualization of planning variants could be integrated in a visualization to provide valuable 3D applications. Further, 2D maps can be applied to the 3D digital terrain model (DTM). Both, extra 3D model data and terrain textures can be fetched from external services. For texturing of the DTM, map data is requested on demand from a Web Map Service. To optimize the server's response times, caching strategies are applied in order to optimize the number of requests to be processed by the external WMS and therefore to optimize rendering performance of the server.

Additional 3D content can be integrated either by using the preprocessing service to create an out-of-core data structure optimized for rendering or, in the case of smaller data sets, directly into the visualization process of the rendering server. The additional contents for map generation can be fetched from a W3DS or loaded from files. The server is configured with a set of 3D models using a configuration file on startup. Additional contents can be loaded during runtime. This allows a fairly flexible image generation of a broad range of 3D content. Since this type of data integration only demands server-side effort, client applications using the service can easily work with new, updated or altered data. While updated data does not need any changes on client-side, additional data sets need an additional parameter for client requests. This has to be considered when clients should be able to request additional datasets from the rendering service.

Server Side Rendering

The 3D rendering service is responsible for scene management and image synthesis. It accesses the optimized 2D and 3D geodata, constructs and optimizes the corresponding scene graph, and implements the core 3D rendering algorithms. State-of-the-art 3D rendering techniques for 3D models are characterized by a strong use of the programmable rendering pipeline. To achieve high-quality results in real-time, implementations are typically based on tightly dependent shader programs, scene graph representations, spatial data structures, level-of-detail techniques, and multi-pass rendering techniques. The implementation of such techniques does heavily rely on the underlying 3D rendering hardware and software, which make their development for a broad range of client platforms and hardware a costly and time consuming task. In contrast, the dedicated rendering server in our approach provides a controlled environment for the implementation of 3D rendering techniques. This leads to a consistent quality of the generated views, regardless of a client's hardware and software. Moreover, it leads to accelerated development and distribution of 3D web mapping applications as well as an increased maintainability of such systems. Further, the server system represents a single point of update, i.e., updated 3D content can be delivered to all clients, without altering client implementations or redistributing geodata to client applications.

Stylization of 3D Contents

Stylization of 3D maps is one of the main issues of the 3D map generation process. It supports the efficient communication of geoinformation. In our 3D rendering service, we support flexible stylization of 3D map contents. For this it provides a number of options to style the 3D maps. Specific textures can be applied to each model element, e.g., a terrain model can be textured using different 2D maps. Further, different 3D rendering effects can be applied per model element, e.g., a specialized rendering for planned constructions. Additionally, the 3D rendering service offers so called image styles that affect the overall appearance of the 3D scene.

This stylization is implemented as an image-based post-processing executed after the 3D rendering of the scene geometry and textures has been performed. Data from different image layers is used together with additional configuration options to configure the image stylization effect, e.g. the id of a scene object to be highlighted by a halo effect. Stylization effects are implemented efficiently using the graphics hardware. Unnecessary copy operations of source image-layers are avoided by reusing them on the graphics card without prior download into the main memory. This way, the graphics hardware of the rendering server is used in a very efficient way for image-based computations, e.g., non-photorealistic rendering. Examples for image-based styling are depicted in Figure 4.

The image-based stylization has one major advantage compared to conventional techniques for 3D-stylization: The computational complexity does not depend on the model size and complexity. The computational costs are mainly dependent on the resolution of the source and target images and the complexity of the desired effect. Image-based techniques can also be used to visualize thematic data, e.g. through projection of image data or applying color values for certain objects encoding specific data values.

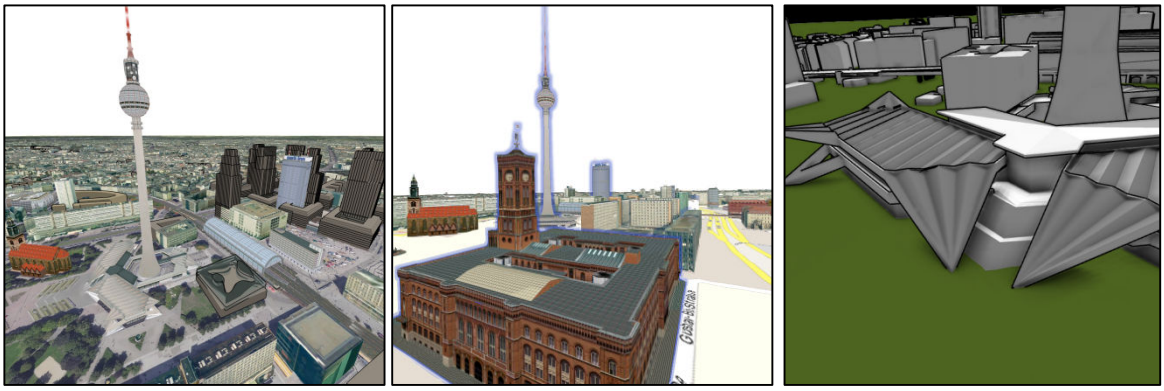

Figure 4: Examples for image-based stylization techniques available in the 3D rendering service. From left to right: Different model elements are styled independently, a halo effect highlights selected features, to increase spatial perception a global illumination effect is applied.

Client Applications

In this section we will provide an overview over of client implementations: (a) An iOS application providing a fully interactive user experience and (b) a very lightweight, browser based client implemented in JavaScript, allowing a stepwise camera interaction. These clients are designed as proof of concept to show, that 3D maps can be provisioned interactively on devices either with limited or unknown hardware and network capabilities or in restricted software environments, such as a web browser.

The interactive mobile client for iOS devices, namely iPad, iPhone, and iPod touch, uses color, depth, and object ID information retrieved as image data from the 3D rendering service to reconstruct the server-side 3D model. The 3D client basically displays the textured cube map geometry, with the initial camera view point in the cube center. It demands only moderate 3D graphics capabilities from the client device: The six-sided cube requires 12 triangles and corresponding six 2D textures (e.g., 512x512 texel, depending on display size of the target device). The user

navigation and interaction operates always on the most recent G-buffer cube map received. If the camera is moved or zoomed, a request for a new cube map is filed, and the current cube map is marked as outdated and an updated one is fetched from the server. Still, navigation and interaction can operate on the current cube map, independently of the provision of the requested new cube map. The artifacts that tend to occur include insufficient image resolution (i.e., blurring effects) and incorrect 3D object visibility. The client-side 3D rendering process can be enhanced by various image-based rendering techniques. Most importantly, a cube map's depth layer can be alternatively encoded as a depth mesh (Pajarola et al., 2004), which represents a triangulation of the depth image. This way, depth meshes are "directly enabling fast intermediate view rendering" (Farin et al., 2007), using the depth mesh instead of the cube map geometry; efficient algorithms for compression and meshing exist (Sarkis et al. 2010). The client switches to a depth mesh scene representation as soon as the position of the virtual camera changes. However, if a depth mesh is not displayed from the reference view, usually holes or rubber-sheets become visible, which can be avoided by requesting and rendering additional depth meshes at the cost of performance and utilized network bandwidth. Also the graphic requirements for this type of 3D scene reconstruction are very modest for mobile devices. To provide an omni-directional scene reconstruction, currently a maximum of six meshes each containing maximum 65536 vertices is rendered. The vertices are textured using the available textures from the six sides of cube map.

The browser-based JavaScript client is a very lightweight front-end for 3D maps. It provides a stepwise navigation metaphor through overlaid UI elements for manipulating the camera positions. Further, additional UI-Elements exist to customize the current style and 3D model selection. Each input action, e.g. a mouse click, updates the image in this web-page. This leads to a less interactive user experience, but still allows navigating through the 3D map with a minimum of hardware and software requirement. Since the client exclusively displays images and manipulates the HTML-DOM, it is able to run and to deliver exactly the same visualization result on every browser that supports Javascript.

Figure 5: iPad application running the virtual 3D city model of Berlin using a WVS for image generation (left) and browser-based Javascript client supporting a stepwise interaction (right).

Case Study – Berlin 3D

To verify our system design we conducted a case study together with the Berlin Partner GmbH, working in the area of city marketing for the city of Berlin and using a virtual 3D city model of Berlin to present and promote available properties to potential investors. The virtual 3D city model of Berlin (more than 500,000 fully textured building models, 350 of them in CityGML LOD 3 or 4) is one of the largest city models in the world.

The goal of this collaboration was to bring high quality visualization of the Berlin virtual 3D city model to mobile devices. Our preprocessing service was used to organize and optimize the model. We integrated the preprocessed data into a rendering service instance running on a remote server (2.7 GHz Intel Xeon, quad core, 16 GB RAM, Geforce GTX 460 2 GB graphics memory). The 3D client application using the 3D rendering server was implemented for Apple iOS device, such as iPad or iPhone. It provides a native, touch-based user interface to control the virtual 3D camera and to select and style 3D contents. Intuitive touch gestures allow even inexperienced users to efficiently explore the 3D scene (Kin et al., 2009). The touch interface for camera interaction supports camera movement in three dimensions and also the change of camera orientation. The application assists users in controlling the 3D camera by applying constraints for camera parameters in order to prevent users from using disorienting camera configurations (Buchholz et al., 2005).

The service-oriented 3D rendering approach can reduce cost and time intensive efforts, resulting from, e.g., physically moving a 3D desktop system including its 3D hardware and the data storage to different locations. For example, instead of using a single view on a large screen, participants can have their own views on their tablets.

Conclusions and Future Work

We presented two examples for client applications running in different, restricted environments. Both clients use the available resources efficiently to provide access to 3D maps, based on massive amounts of complex 3D geodata, which was hard to achieve in a good visual quality beforehand. By lowering the entry barriers in terms of hardware and software prerequisites, but also concerning user interaction, 3D maps can become useable by a broader audience. Our case study demonstrated that our approach and platform can be used for interactive 3D web mapping even of large-scale 3D contents, such as a large virtual 3D city model. The system was well accepted by users. Especially, they reported that the client's touch-based interface offers easy to use means for the navigation of complex virtual 3D spaces. The service-based system for 3D web mapping presented in this paper, together with lightweight clients, offer various opportunities for the integration of 3D visualization components into existing IT and geodata infrastructures.

Acknowledgements

We thank Berlin Partner GmbH as well as the 3D Content Logistics GmbH for the collaboration on this project.

References

Altmaier, A. & Kolbe, T.H., 2003. Applications and Solutions for Interoperable 3D Geo-Visualization. In *Proceedings of the Photogrammetric Week 2003*. Stuttgart: Wichmann, pp. 251–267.

Basanow, J. et al., 2008. Towards 3D Spatial Data Infrastructures (3D-SDI) based on open standards – experiences, results and future issues. In *Advances in 3D Geoinformation Systems*. Springer, pp. 65–86.

Bishr, Y., 1998. Overcoming the semantic and other barriers to GIS interoperability. *Int. Journal of Geographical Information Science*, 12(4), pp.299–314.

Boukerche, A. & Pazzi, R.W.N., 2006. Remote rendering and streaming of progressive panoramas for mobile devices. In *Proceedings of the 14th ann. ACM Int. Conf. on Multimedia - MULTIMEDIA '06*. New York, New York, USA: ACM Press, p. 691.

Buchholz, H., Bohnet, J. & Dollner, J*., 2005. Smart and Physically-Based Navigation in 3D Geovirtual Environments. In Ninth International Conference on Information Visualization (IV'05). IEEE, pp. 629–635.*

Farin, D., Peerlings, R. & de With, P.H.N., 2007. Depth-Image Representation Employing Meshes for Intermediate-View Rendering and Coding. In *2007 3DTV Conference*. Ieee, pp. 1–4.

Gröger, G., Kolbe, T. H., Nagel, C., Häfele, K., 2012. *OpenGIS® City Geography Markup Language (CityGML) Encoding Standard Version 2.0.0*, Available at: http://www.opengeospatial.org/standards/citygml.

Haber, R.B. & McNapp, D.A., 1990. Visualization Idioms: A Conceptual Model for Scientific Visualization Systems. In *Visualization in Scientific Computing*. IEEE, pp. 74–93.

Hagedorn, B., 2010. Web view service discussion paper, Version 0.6. 0. *Open Geospatial Consortium Inc*.

Hildebrandt, D., Hagedorn, B. & Döllner, J., 2011. Image-based strategies for interactive visualization of complex 3D geovirtual environments on lightweight devices. *Journal of Location Based Services*, 5(2), p.100-120.

Kin, K., Agrawala, M. & DeRose, T., 2009. Determining the benefits of direct-touch, bimanual, and multifinger input on a multitouch workstation. In *Proceedings of Graphics Interface 2009*. Canadian Information Processing Society, pp. 119–124.

Koller, D., Turitzin, M. & Levoy, M., 2004. Protected interactive 3D graphics via remote rendering. *... on Graphics (TOG)*. Available at: http://dl.acm.org/citation.cfm?id=1015782 [Accessed August 23, 2012].

Lamberti, F. & Sanna, A., 2007. A streaming-based solution for remote visualization of 3D graphics on mobile devices. *IEEE transactions on visualization and computer graphics*, 13(2), pp.247–60.

Pająk, D., Herzog, R., Eisenmann, E., Myszkowski, K. and Seidel, H., 2011. Scalable Remote Rendering with Depth and Motion-flow Augmented Streaming. *Computer Graphics Forum*, 30(2), pp.415–424.

Pajarola, R., Sainz, M. & Meng, Y., 2004. Dmesh: Fast depth-image meshing and warping. *International Journal of Image and Graphics*, 4(4), pp.1–29. Available at: http://www.ifi.uzh.ch/vmml/publications/older-puclications/DMesh.pdf [Accessed August 24, 2012].

Paravati, G., Sanna, A., Lamberti, F. and Ciminiera, L., 2011. An open and scalable architecture for delivering 3D shared visualization services to heterogeneous devices. *Concurrency Computat.: Pract. Exper. 2011; 23:1179–1195*

Portele, C., 2007. OpenGIS Geography Markup Language (GML) Encoding Standard, Version 3.2.1. Available at: http://www.opengeospatial.org/standards/gml.

Sarkis, M., Zia, W. & Diepold, K., 2010. Fast depth map compression and meshing with compressed tritree. In *Computer Vision–ACCV 2009*. pp. 44–55.

Schilling, A. & Kolbe, T.H., 2010. Draft for Candidate OpenGIS® Web 3D Service Interface Standard, Version 0.4.0.

Schut, P., 2007. OpenGIS® Web Processing Service, Version 1.0.0. Available at: http://www.opengeospatial.org/standards/wps.

Vretanos, P.A., 2005. OpenGIS Web Feature Service (WFS) Implementation Specification. Available at: http://www.opengeospatial.org/standards/wfs.

Wessels, A. Purvis, M., Jackson, J., Rahman, S., 2011. Remote Data Visualization through WebSockets. *2011 8th Int. Conf. on Inf. Technology: New Generations*, pp.1050–1051.